MANUAL FORTRAN 77



ALPHA Ltd. @ 1991

Tipografia MIRTON
1900 TIMISOARA
Str. Samuil Micu nr.7
Telefon 96 - 18.35.25.

C U P R I N S

MIRA SOFTWARE ** COMPILATOR FORTRAN
MANUAL DE UTILIZARE

1. GENERALITATI

Compilatorul FORTRAN produs de firma MIRA SOFTWARE pentru calculatorul SINCLAIR SPECTRUM se bazeaza pe FORTRAN77S care este un subset din FORTRAN77 STANDARD.

El compileaza programe scrise în FORTRAN, transformîndu-le în programe în cod masina, care pot fi apoi rulate INDEPENDENT DE COMPILATOR!

Acest manual este împartit în două parti:

Prima parte se refera la încarcarea si la utilizarea compilatorului, iar a doua parte contine detalii asupra particularitatilor limbajului FORTRAN implementat în compilator.

Pentru a începe programarea trebuie să încărcati (în mod normal cu comanda LOAD) prima parte a compilatorului, denumit **FORTRAN1** (precedat de un bloc BASIC denumit LOADER).

Cînd acesta s-a încarcat normal, în partea stînga / jos a ecranului apare mesajul: '**Press Required Key**' (apasati tasta necesara). Cu aceasta editorul este pus în functie, deci prin apăsarea anumitor taste puteti scrie sau manipula programul.

2. INTRODUCEREA LINIILOR DE PROGRAM

Apasînd ENTER se va trece editorul în modul de lucru pentru introducerea liniilor de program FORTRAN în orice succesiune.

Limbajul FORTRAN necesita un format specific pentru liniile de program: linia nu poate fi mai lungă de 80 de caractere. Primul caracter poate fi c sau **m** numai pentru o linie de comentarii, care va fi ignorata de compilator, sau **D** pentru o linie care va fi compilata si utilizata numai pentru depanarea programului si care poate fi o linie de comentariu sau de program. Daca primul caracter este diferit de cele specificate anterior, atunci primele 5 caractere trebuie lansate blank (spatii), fie sa contina o eticheta formata dintr-un numar de maxium cinci cifre; al 6-lea caracter trebuie sa fie lăsat blank pentru o linie de program. Caracterele de la pozitia 7 la 80 inclusiv sînt utilizate pentru instructiunile FORTRAN.

Retineti că în instructiunile FORTRAN se folosesc numai caractere MAJUSCULE si că în fiecare linie este permisă numai o singură instructiune. Dacă instructiunea este prea lungă, aceasta poate fi continuată pe linia următoare punînd + ca al 6-lea caracter din linia de continuare si continuînd instructiunea începînd cu pozitia a 7-a.

Cînd o linie este introdusă, compilatorul face o verificare sumară a sintaxei înainte de a introduce cu adevarat linia în program. Dacă este detectată o eroare atunci este afisat semnul '?' în pozitia probabilă a erorii si se asteaptă corectarea.

Pe masură ce liniile sînt introduse în program, cursorul va marca pozitia ultimei linii introduse. In listing fiecare linie are un număr de ordine (în dreapta pe ecran), care indică pozitia

acesteia în program. Aceste numere nu trebuie confundate cu numerele atribuite etichetelor, care pot să ocupe numai primele 5 caractere ale liniei.

Pentru a iesi din modul de lucru pentru introducerea liniilor programului, apăsati concomitent CAPS SHIFT si 6. In partea de jos a ecranului vor apărea mesajele: STOP IN INPUT si PRESS REQUIRED KEY.

3. TASTELE DE CONTROL

Cînd programul este listat pe ecran, cursorul '>' pulsator marcheaza pozitia liniei curente. Acest cursor poate fi mutat în sus si în jos apasînd clapele 7 sau 6. Dacă odată cu una din acestea se apasă si CAPS SHIFT, atunci cursorul sare cu cîte 8 linii odată.

Linia curentă poate fi editată apăsînd tasta 1 sau EDIT, după care ea va fi copiată în cîmpul de redactare. După eventualele modificări se apasă ENTER (ca la o linie nouă). Apăsînd 'E' se poate introduce o linie înainte de linia curentă.

Daca după o listare se trece în modul de lucru pentru introducerea liniilor de program, atunci noile linii vor fi introduse după linia curentă.

Apăsînd 'V' puteti lista programul la imprimantă (pe canalul #3).

Pagina de HELP (o lista a comenzilor) se obtine apăsind tasta 'H'.

4. SALVAREA SI INCARCAREA PROGRAMELOR

Apăsînd tasta 'S' se poate salva programul FORTRAN (sursa) pe bandă. Se va cere să introduceti numele programului (max. 10 caractere) si apoi salvarea va decurge după procedura obisnuită. Compilatorul va cere ca programul sa fie verificat (VERIFY), operatie care se va desfasura la fel ca de obicei.

SE RECOMANDA CA PRORAMUL SURSA SA FIE TOTDEAUNA SALVAT PE BANDA INAINTE DE A FI COMPILAT, caci dupa compilare el este sters din memorie.

Pentru a încărca de pe bandă programul-sursă se apasă 'J', după care se cere să introduceti numele programului. Dacă în locul numelui apăsati doar ENTER, se va încarcă primul program întîlnit pe bandă. Dacă în calculator se va afla deja încărcat un program, noul program va fi încărcat asamblat cu vechiul program începînd de la linia curentă (marcată de cursor).

5. DEPLASAREA UNUI BLOC DE LINII

Este posibil să deplasati un bloc de linii în cadrul programului astfel:

Duceti cursorul la linia de început a blocului si marcati acest lucru apăsînd tasta 'B', apoi introduceti cursorul DUPA ultima linie a blocului si apăsati tasta 'K'. Cu aceasta blocul este marcat si dacă doriti numai stergerea sa din program este suficient să comandati DELETE. Daca doriti deplasarea blocului în program pozitionati cursorul la locul unde doriti să înceapă blocul marcat si apăsati tasta 'R'.

6. COMPILAREA PROGRAMELOR

Un program odată introdus (cu ajutorul editorului), poate fi compilat apăsînd tasta 'X'. Compilatorul va face o verificare sintactică amănunțită a programului; dacă este detectată o eroare, cursorul va fi plasat la prima linie cu probleme, iar pozitia probabilă a erorii va fi indicată prin semnul '?'. Acum comandînd EDIT se pot face corecturile cuvenite. Uneori în partea de jos a ecranului poate apărea un mesj care indică natura erorii.

Pe masură ce compilarea are loc, în memoria video (cea destinată DISPLAY-ului) vor fi plasate datele programului: tabela numerelor de variabile, etc. Pe ecran va apărea o imagine haotică si dacă nu mai sînt detectate erori, cînd se termină compilarea se va auzi un semnal sonor, care vă atentionează că trebuie să încărcati partea a doua a compilatorului.

Acesta este denumit 'CODER', dar denumirea sa nu va apărea pe ecran deoarece ar afecta datele depuse acolo de prima parte a compilatorului. Pentru a încarca a doua parte a compilatorului nu este nevoie de o comanda anume, fiind suficient să porniti banda. Dacă aceasta se va încărca cu eroare, se va auzi un semnal sonor si trebuie să repetati operatia de încărcare a acestei părti. Dacă încărcarea a decurs normal, compilarea continua imediat (fără o comanda anume), iar în cazul în care întîmplător este detectata o eroare, se va auzi un semnal sonor SI VA FI AFISAT NUMARUL DE COD AL ERORII.

In acest caz este necesar sa reluati totul de la început folosind editorul (prima parte a compilatorului) si programul sursă salvat pe caseta, în care veti căuta sa corectati eroarea (care a împiedicat a doua parte a compilarii).

Cînd compilarea s-a terminat cu succes, pe ecran si la printer (canal ?3) vor apare anumite informatii ca în exemplul următor:

CLEAR 60056....> reprezintă adresa maxima pentru RAMTOP cînd se utilizează programul compilat.

SAVE 60075....> adresa de început al blocului în cod masină care este programul obiect.

LENHT 5293....> lungimea în Bytes a programului obiect.

REM 60075....> adresa blocului 'COMMON BLANK' al programului

Dacă nu aveti conectată o imprimantă la calculator, trebuie să vă notati aceste date ABSOLUT NECESARE PENTRU MANIPULAREA ULTERIOARA APROGRAMULUI OBIECT, căci la prima apasare pe tasta, aceasta împreună cu compilatorul vor fi sterse din memorie, rămînînd numai codul obiect la adresa si cu lungimea mentionate. Inainte de a încărca acest bloc trebuie sa pozitionati RAMTOP-ul la valoarea aratăta (comandînd CLEAR urmat de numărul respectiv).

Codul obiect poate fi lansat în executie cu comanda 'RANDOMIZE USR 63500'. El poate fi utilizat în cadrul unui program scris în BASIC. In acest caz dacă pe timpul rulării programului FORTRAN compilat va apare o eroare, va fi afisat mesajul respectiv de eroare IMPREUNA CU NUMARUL LINIEI BASIC IN CARE S-A APELAT CODUL OBIECT FORTRAN.

7. COMENZI FORTRAN

In acest capitol sînt prezentate detaliile asupra comenzilor FORTRAN implementate precum si informatii asupra deosebirilor fatà de FORTRAN STANDART. Dacà nu sîntati initiat, nu va fi suficient acest material pentru a va învàta programarea în FORTRAN.

In acest caz este recomandabil sà folositi un manual de introducere în FORTRAN la un nivel potrivit cu pregàtirea dvs.

7.1. TIPURI DE DATE

Tipurile de date acceptate de acest compilator sînt: INTEGER, REAL, LOGICAL SI CHARACTER. Tipurile COMPLEX si BOUBLE PRECISION ce se utilizeazà în FORTRAN77 nu sînt implementate.

Valorile întregi trebuie sà fie în intervalul (-32767... +32767). In timp ce valorile reale sînt tratate în acelasi mod ca în SPECTRUM BASIC. Constantele reale trebuie sà continà fie un punct zecimal si/sau un exponent. De ex. 300 este o constantà întreagà, pe cînd 300. sau 3e2 sînt constante reale iar 3000000 va fi considerata constantà întreagà si deci tratata ca eroare!

Cele douà constante logice sînt .TRUE. si .FALSE, (atentie la punctele de la început si de la sfîrsit!).

Constatele alfa numerice (CHARACTER) sînt marcate la început si la sfîrsit de un apostrof. Apostroful inclus în sir va fi dublat.

7.2. VARIABIE SI DECLARATII DE TIPURI

Numele de varabile si alti indicatori în FORTRAN nu trebuie sà fie mai lungi de 6 caractere, din care primul trebuie sá fie o literà iar urmàtoarele pot fi cifre sau litere.

Fiecare variabilà apartine unuia dintre tipurile mentionate si care este determinat fie printr-o declaratie de tip fie implicit. Declaratia de tip consta din deumirea tipului (INTEGER, REAL, LOGICAL sau CHARACTER), urmat de numele variabilelor. La variabilele de tip CHARACTER lungimea acestora poate fi specificatà, de ex.:

```
CHARACTER*6 A,B*3,C
```

atribuie lungimea 6 variabilelor A si C iar variabilei B lungimea 3. In implementarea pe care o prezentàm, variabilele alfanumerice pot avea lungimi de maximum 255 de caractere.

Dacà o variabilà nu apare într-o declaratie de tip atunci tipul se stabileste de la sine astfel: daca numele începe cu unul dintre caracterele I, J, K, L, M sau N, atunci va fi consideratà de tip INTEGER iar în caz contrar de tip REAL. Aceastà conventie poate fi modificatà prin declaratia IMPLICIT care specificà un tip de variabile urmat de domeniul literelor la care se aplicà, de ex.: IMPLICIT CHARACTER*5 (A-D), LOGICAL (L), INTEGER (X-Z) (cu alte cuvinte conventia anterioarà referitoare la variabilele INTEGER este echivalentà cu IMPLICIT INTEGER (I-N)).

7.3. EXPRESII SI ATRIBUIRI

O instructiune de atribuire are forma:

nume variabilă = expresie

unde rezultatul evaluării expresiei trebuie să fie de acelasi tip cu variabila sau să fie un numar întreg, daca variabila e declarată ca fiind reală. O expresie aritmetica (REAL sau INTEGER) este compusă din diferiti termeni combinati fiecare combinati cu paranteze '(' sau ')' si/sau cu operatori '+', '-', '*', '/' (unde A**B reprezintă A ridicat la puterea B).

Termenii trebuie să fie constante, variabilele, sau rezultatul apelului unor functii (intrinseci sau definite de utilizator).

Dacă o valoare apare acolo unde este cerută o valoare reală, sau invers, atunci aceasta poate fi convertită în mod automat în timpul corespunzator.

In continuare este prezentată lista functiilor interseci disponibile:

(X si Y reprezintă valorile reale, iar I si J valori întregi)

DENUMIREA	DEFINITIE
SQRT(X)	Rădăcina pătrată.
EXP(X)	Exponentială.
ALOG10(X)	Logaritm în baza 10.
ALOG(X)	Logaritm în baza e (logaritm natural).
SIN(X)	Sinus.
COS(X)	Cosinus.
TAN(X)	Tangentă.
ASIN(X)	ArcSinus.
ACOS(X)	ArcCosinus.
ATAN(X)	ArcTangentă.
ATAN2(X,Y)	ArcTang(X/Y) în cadranul corect!
SINH(X)	Sinus hiberbolic
COSH(X)	Cosinus hiberbolic
TANH(X)	Tangentă hiberbolică.
AINT(X)	Trunchiere în jos la întreg, dar rezultatul este de tip REAL.
AINT(X)	Rotunjire la cel mai apropiat întreg, dar rezultatul este de tipul REAL.
ABS(X)	Valoarea absolută a lui X
AMOD(X,Y)	Restul împărtirii X/Y
SIGN(X,Y)	ABS(X) cu semnul lui Y.
DIM(X,Y)	X-Y daca X>Y, altfel 0 (zero).
INT(X)	Trunchiere la întreg, rezultatul INTEGER.
FIX(X)	Similar cu INT(X)
NINT(X)	Rotunjirea la cel mai apropiat întreg, (rezultatul de tip INTEGER).
REAL(I)	Convertire la tipul REAL.
FLOAT(I)	Similar cu REAL(I).
IABS(I)	Valoarea absolută a lui I.
MOD(I,J)	Restul împărtirii I/J.
ISIGN(I,J)	IABS(I) cu semnul lui J.
IDIM(I,J)	I-J dacă I>J, altfel 0 (zero).

AMAX1(X1, X2,....,Xn)	Valoarea maximă din setul de variabile dat.
MAX1(X1, X2,....Xn)	Partea întreagă a valorii maxime din setul de variabile citat în paranteză.
AMIN1(X1, X2,....,Xn)	Valoarea minimă din setul de variabile dat.
MIN1(X1, X2,....,Xn)	Partea întreagă a valorii minime din setul de variabile citat în paranteză.
AMAXO(I1, I2,....,In)	Valoarea maximă convertită la tipul REAL .
MAXO(I1, I2,....In)	Valoarea maximă.
AMINO(I1, I2,....In)	Valoarea minimă convertită la tipul REAL.
MINO(I1, I2,....In)	Valoarea minimă.

Expresiile logice constau din termeni logici combinati cu paranteze si/sau cu operatori logici: .NOT., .AND., .OR., .EQV., .NEQY. (atentie la punctele de la început si de la sfîrsit!). Termenii logici pot fi variabile logice, functiuni sau constante si pot fi de asemenea expresii relationale. Acestea din urmă constau din compararea a două expresii aritmetice utilizînd operatori relationari: .LT., .LE., .EQ., .NE., .GT., .GE. (atentie la punctele de la început si de la sfîrsit!). Retineti ca simbolurile uzuale:<,>,<=,>=,<> NU SE FOLOSESC IN FORTRAN IN ALCATUIREA EXPRESIILOR LOGICE.

Expresiile alfanumerice (literele) pot fi la rîndul lor constante sau variabile. IN ACEASTA VERSIUNE DE FORTRAN FUNCTIUNILE "SUBSTRING" SI "CONCATENARE" PENTRU MANIPULAREA SIRURILOR DE CARACTERE NU SINT ACEPTATE.

7.4 STRUCTURA PROGRAMELOR SI INSTRUCTIUNI DE COMANDA

Un program FORTRAN constă dintr-un număr de unităti de program, una dintre ele este programul principal, celelalte fiind subrutine si/sau functii scrise de utilizator.

Programul principal poate începe, optional, cu declaratia PROGRAM urmată de numele programului. Ultima instructiune a fiecărei unităti de program este END, care va cauza revenirea la programul apelant, daca unitatea este o subrutină, sau revenirea în BASIC, dacă este vorba de programul princpal.

Instructiunea STOP va cauza revenirea directa în BASIC.

Cuvîntul cheie STOP poate fi urmat de un sir de caractere sau de un număr care va fi afisat la executia instructiunii. Instructiunea PAUSE va opri executia proramului FORTRAN pînă cînd este apasată o tastă. Si această instructiune poate fi urmată de un sir de caractere care vor fi afisate la executia instructiunii.

Există trei feluri de instructiuni IF în FORTRAN: blocuri IF, IF logic si IF aritmetic.

Blocul IF constă dintr-un cuvînt cheie IF urmat de o expresie logică inclusă între paranteze, urmat de cuvîntul cheie THEN ....

Fiecare instructiune 'bloc IF' trebuie să se termine cu o instructiune END IF. Sînt admise de asemenea instructiunile ELSE si ELSE IF.

Instructiunea IF logic este constituită sub forma unei singure instructiuni, compusă din IF urmată de o expresie logică închisă în paranteze si apoi de o instructiune care se va executa în cazul în care expresia logică este adevarată.

Instructiunea IF aritmetic este urmat de o expresie aritmetica inclusă în paranteze, si apoi de 3 etichete: dacă valoarea expresiei va fi negativă, atunci executia programului va fi trensferată primei etichete, daca este 0 celei de a doua etichete, iar daca este pozitivă celei de a treia.

Buclele în FORTRAN sînt implementete utilizind bucle DO. Instructiunea are forma:

DO eticheta IVAR = IE1, IE2, IE3

unde 'eticheta' este eticheta care identifica ultima instructiune din bucla, 'IVAR' este variabila de tip INTEGER , iar IE1, IE2, IE3 sînt expresii INTEGER care precizează respectiv valoarea initială a lui IVAR, valoarea finală si pasul. Bucla este executata de INT ((E2-E1+E3)/E3) ori si nu se executa niciodată daca IVAR <=0.

Exista 3 tipuri de instructiuni GOTO în FORTRAN:

Instructiunea GOTO neconditionat, care constă din cuvîntul cheie GOTO, urmat de eticheta unei instructiuni unde trebuie transferată neconditionat executia programului.

Instructiunea GOTO calculat, este compusă din cuvîntul cheie GOTO urmat de o secventă de etichete incluse între paranteze si apoi de o expresie de tipul INTEGER. Executia programului este transferata la eticheta cu locul în secventă egal cu valoare expresiei de tip INTEGER (de la sfîrsitul instructiunii)

Instructiunea GOTO asignat constă în cuvîntul cheie GOTOn urmat de o variabilă de tip INTEGER si apoi de o listă de etichete inclusă în paranteze. Variabilei trebuie să i se fi atribuit anterior valoarea uneia dintre etichete printr-o instructiune ASSIGN de forma:

ASSIGN eticheta TO IVAR

La executia instructiunii GOTO asignat, daca variabilei 'IVAR' i s-a atribuit valoarea uneia din etichetele din paranteze (din componenta instructiunii), atunci executia programului va fi transferată la instructiunea care poartă aceasta etichetă.

7.5. MEMORIA VARIABILELOR

Ca si declaratiile de tip descrise anterior, FORTRAN dispune de instructiuni care controlează plasarea variabilelor în memorie. Acestea nu sînt instructiuni executabile si apar în program înaintea oricăror instructiuni executabile.

Tablourile de variabile de oricare din cele 4 tipuri pot fi declarate cu instructiunea DIMENSION, care are forma:

DIMENSION AR(20,10),NAME (30)

Prin aceasta s-au declarat tablourile AR-bidimensional (20*10 elemente) si NAME-unidimensional cu 30 elemente. Un anumit element al unui tablou este adresat într-o expresie specificînd pozitia elementului - de ex:

AR(2,3)

De retinut că în această implementare a limbajului FORTRAN, elementul este considerat pozitionat în limitele TOTALE ale tabloului si nu în limitele fiecărei dimensiuni! Pentru clarificare, în

exemplul anterior o referire AR(25,3) nu va fi considerată eroare pentru că prima dimensiune ar fi depasită (20), ci se va referi la elementul AR(5,4).

Declaratia unui tablou poate fi deplasată într-o declaratie de tip sau într-o instructiune COMMON.

Variabilele si tablourile pot fi initializate prin instructiunea DATA, care are forma:

```
DATA var1, var2,.../varİ1, var12,.../
```

unde var1, var2, etc. reprezintă numele, variabilelor, tablourilor sau elementelor, iar var11, var12, etc. sînt valorile atribuite variabilelor si/sau tablourilor în ordinea în care apar în prima parte a instructiunii. Notatia: numar*valoare poate fi utilizată în lista de valori pentru a arăta ca valoarea se repetă de un numar de ori. Liste 'DD IMPLICITE' în instructiunea DATA nu sînt implementate în compilator.

De asemenea nu este implementată instructiunea PARAMETER.

Blocurile COMMON reprezintă un mod de a indica faptul ca memoria afectată unor variabile este utilizată în comun de mai multe unitati de program. Instructiunea COMMON are forma tip:

```
COMMON/NAME/VAR1,VAR2,...
```

care indică faptul ca variabilele VAR1, VAR2, etc. vor fi grupate într-un bloc comun numit NAME si pot fi acceptate de către altă unitate de program care contine o instructiune similara. Dacă numele blocului va fi omis, atunci variabilele vor fi grupate în blocul COMMON BLANK.

Variabilele care sînt initializate prin instructiunea DATA nu trebuie sa fie memorate într-un bloc COMMON.

Două sau mai multe variabile pot fi făcute să ocupe acelasi spatiu în memorie prin instructiunea EQUIVALENCE, care are forma:

```
EQUIVALENCE (A,B,C (4)).
```

Aceasta indică faptul că variabilele (sau string-urile) A si B vor ocupa aceeasi positie (început) ca si elementul C(4) al stringului C(?).

Retineti faptul că limbajul FORTRAN permite ca aceleasi variabile să apară în mai multe instructiuni EQUIVALENCE, ca si într-o instructiune COMMON. Compilatorul va aloca optim memoria în functie de aceste declaratii. Evident amestecul si suprapunerea variabilelor de diferite tipuri trebuiesc evitate. Totusi, dacă doriti să utilizati acelasi spatiu de memorie pentru diferitele tipuri de date, trebuie să aveti în vedere spatiul ocupat de fiecare tip, care în aceasta implementare este de 5 bytes pentru o valoare reală, 2 bytes pentru o valoare întreagă, 1 byte pentru caractere si valori logice (într-o variabilă simplă), iar pentru un tablou logic (?) sînt memorate 8 valori logice/Byte.

Aceasta diferă fată de forma standard de FORTRAN, care specifică faptul că valorile REAL, INTEGER si LOGICAL trebuie să ocupe acelasi spatiu în memorie.

Inceputul blocului COMMON BLANK în memorie este dat de numă-

rul care urmează după REM în datele afisate la sfîrsitul compilării si vă permite àa accesati variabilele FORTRAN din BASIC.

7.6. FUNCTII SI SUBRUTINE

In afară de programul principal, un program FORTRAN poate să contină alte unităti de program numite subprograme.Acestea pot fi functii sau subrutine. Prima instructiune dintr-un subprogram trebuie să fie instructiunea SUBROUTINE sau FUNCTION, după care urmează numele subprogramului si apoi cuprinse între paranteze, lista argumentelor. De retinut că la instructiunea FUNCTION parantezele trebuie să existe chiar dacă nu sînt argumente! O functie (FUNCTION) apartine unui tip declarat explicit prin cuvîntul INTEGER, REAL sau LOGICAL care poate precede cuvîntul cheie. FUNCTION sau după regulile tipului IMPLICIT. In această implementare nu este necesar să declarati tipul unei functii în fiecare unitate de program unde aceasta este apelată. Un subprogram-functie trebuie să contină o instructiune de asignare prin care se asignează valoarea (returnată) a functiei. Apelarea unui subprogram-functie se obtine prin includerea numelui (si a argumentelor respective) în cadrul unei expresii unde va furniza si valoarea cerută.

O subrutina (SUBROUTINE) este apelată în altă unitate de program prin instructiunea CALL, ca în exemplul:

```
CALL GRAPH (A, B, X+1)
```

De retinut că atît în cazul unitătilor FUNCTION cît si SUBROUTINE, daca argumentul transmis este o variabilă, un tablou sau un element de tablou ,acesta va fi transmis prin referire. Adică valoarea argumentului poate fi modificată de către subrutină (sau functie), iar dacă argumentul este o expresie, atunci valoarea argumentului se va obtine prin evaluarea acestei expresii. Tipul argumentelor curente trebuie să corespundă cu cel al parametrilor din lista de parametrii a subrutinei. Dacă un argument este întreg si altul real, atunci se va face conversia necesară, dar argumentul va fi transmis prin valoare si nu prin referire.

NOTA Este posibil de transmis o matrice cind este necesară o singură variabilă , dar atunci va fi luat in considerare numai primul element al matricii . Este de asemenea posibil de transmis o singură variabilă sau un element al unei matrici cind este necesară o matrice ; in cazul unei singure variabile, aceasta va fi considerată o matrice de dimensiunea 1 . In cazul unui element al unei matrici , matricea transmisă va fi ceea ce rămine din matricea care urmează elementului specificat.

Cînd unul din argumentele unui subprogram este un tablou, acesta trebuie să fie declarat ca un tablou în cadrul subprogramului. Dimensiunile tabloului pot fi diferite de cele ale tabloului transmis ca argument, atît timp cît dimensiunea totală nu este mai mare decît dimensiunea sa (?).

Este de asemenea posibil ca dimensiunile tablourilor să fie variabile (tablouri ajustabile) si să se specifice ultima dimensiune a tabloului printr-un asterisc, caz în care tabloul va

lua dimensiunea tabloului transmis (tablouri cu dimensiuni implicite).

Subprogramele pot fi de asemenea transmise ca argumente. In acest caz subprogramele curente trebuie să apară intr-o declaratie EXTERNAL (pentru subprogramele definite de utilizator) sau INTRINSIC (pentru subrutine si functii) în programul care apelează, iar argumentele corespunzătoare ale acestora trebuie să apară într-o declaratie EXTERNAL din subprogramul în care acestea apar. Retineti că numărul argumentelor unui subprogram transmis ca argument si tipul argumentelor care apar în programul apelant, trebuie să fie identice, adică INTEGER sau REAL si să nu aibă loc conversii de tip. In continuare se prezintă o listă de subrutine predefinite (intrinsic), care sînt utilizate de acest compilator:

```
CIRCLE (X, Y, R)
DRAW (X, Y)
PLOT (X, Y)
BEEP (X, Y)
ARC (X, Y, R)
```

Toate acestea au argumente reale, iar efectul lor este acelasi ca si în cazul comenzilor cu acelasi nume din BASIC, cu observatia că ARC (X, Y, R) corespunde în BASIC cu DRAW X, Y, Z.

Variabilele din subprograme sînt conservate între apeluri, astfel încît instructiunea SAVE (din FORTRAN) nu este necesară.

7.7. INSTRUCTIUNI DE INTRARE/IESIRE

Instructiunile de intrare/iesire în FORTRAN sînt recunoscute ca unul din punctele 'tari' ale acestui limbaj, deoarece există numeroase căi de a efectua aceasta. Din păcate tocmai în această privintă diferentele între diversele compilatoare sînt cele mai mari, în special în cazul de fată dispozitivele de intrare/iesire din SPECTRUM sînt total diferite fată de tipicul celorlalte calculatoare.
Subsetul de FORTRAN definit ca 'standard' (FORTRAN77S) este mult mai adecvat pentru microcalculatoare, iar compilatorul de fată diferă în unele locuri de FORTRAN77S.

O instructiune de iesire în FORTRAN poate avea forma:

```
WRITE(2, 10) X, Y, (IR(I), I = 3, 30, 3)
```

Aici 2 este numărul unitătii (de iesire) si corespunde unei căi (stream) din BASIC. Astfel 1 se referă în mod normal la introducerea datelor de la claviatură si afisarea datelor în partea de jos a ecranului; 2 afisarea datelor pe ecran; 3 scrierea datelor la im-primanta (printer); alte numere transmit datele către alte canale care trebuie să fie deschise în BASIC înainte de a apela programul FORTRAN compilat. Numărul unitatii (de intrare/iesire) poate fi înlocuit cu un asterisc, specificînd valoarea implicită 1 pentru operatiile de intrare si 2 pentru cele de iesire. Inlocuirea numărului unitàti de intrare/iesire printr-un tablou de caractere nu este permisă în aceasta implemetare.

In exemplul de instructiune de iesire prezentat anterior numărul 10 (al 2-lea număr din prima paranteză) reprezintă

indicatorul de format si se referà la instructiunea FORMAT cu aceasta etichetà:

```
10    FORMAT ('Valorile sînt' , 2f8.2/3x, 10I4)
```

Identificatorul de format poate fi o variabilà de tip INTEGER, care printr-o instructiune ASSIGN sa capete valoarea care constituie eticheta.

Formatul poate fi inclusiv în instructiunea WRITE ca în exemplul urmàtor:

```
WRITE   (2, '(' 'Dimensiunea =' ', 2x, F6.2) SIZE
```

De asemenea indicatorul de format poate fi un asterisc, implicînd o listare într-un format prestabilit, care pentru acest compilator înseamnà cà fiecare valoare este afisatà pe o nouà linie. Daca identificatorul de format este omis, atunci datele de iesire nu vor mai fi formate, adicà vor fi transmise asa cum se gàsesc în memorie.

Compilatorul admite într-o specificatie de FORMAT urmàtoarele coduri:

Iw, A, Aw, Fw.d, Ew.d, Lw.kP. nh..., '...' ,mx, /, virgula.

<u>NU</u> <u>SINT</u> <u>ADMISE</u> (desi în format standard se folosesc):

Iw.n, Fw.dEc, Ew.dEc, Ow.d, Bw.dEc, Tc, TLc, TRc, S, SP, SS, *, BN, BZ.

In lista acestor coduri w, e, n, c, d, m, k sînt de tip INTEGER.

Compilator admite de asemenea codul FORMAT care este urmat de un numar întreg între 0 si 255, determinînd tranmiterea codului ASCII corespunzator si codul CS care va sterge ecranul (unitatea specificata trebuie sa fie 2).

Retineti ca în aceasta versiune a compilatorului nu sînt implementate codurile de control ale imprimantei de la începutul specificatiei de format.

Lista efectiva de intrare/iesire urmareste lista de control a informatiilor. Aceasta poate contine variabile, tablouri, si liste DO I/E implicite (ca mai sus) dar aceasta implementare nu permite aparitia unor expresii în lista I/E. (Astfel sirurile de caractere nu trebuie sa apara într-o lista I/E, ci trebuie sa fie atasate unor variabile, sau tratate prin specificatiile de FORMAT).

O instructiune alternativa este PRINT, în care lista de control a informatiilor este înlocuita de un identificator de format (în mod frecvent un asterisc) si unitatea este stabilita implicit ca fiind 1 - ca în exemplul:

```
PRINT*, A, B, C
```

Instructiunea de intrare în FORTRAN este READ, care poate avea forma fie a instructiunii PRINT, fie a instructiunii WRITE - cu controlul complet al listei de informatii. In cazul al doilea, un specificator END poate fi plasat dupa indicatorul de unitate, permitînd un salt în alta parte a programului în cazul în care este detectat un sfîrsit de fisier la disc, sau la una din ca., cu urmatoarele observatii:

a) Aceasta nu este pozitia obisnnuita pentru specificatorul END din FORTRAN standard.

b) Saltul va fi efectuat numai daca apare conditia de sfîrsit de fisi - dupa un caracter de sfîrsit de linie (CHR$13), în caz contrar în mod normal va apare eroarea 'END of File'.

Alti specificatori de fisier (REC, ERR, IOSTAT) nu sînt implementati.

Introducerea datelor cu format este recomandata si pentru introducerea datelor de la claviatura, deoarece se poate defini clar semnificatia fiecarui caracter. SPECTRUM trateaza toate fisierele ca fisiere cu acces secvential, deci fisiere de tip 'random access files' nu sînt implementate, iar instructiunile: CLOSE, OPEN, INQUIRE, BACKSPACE, ENDFILE si REWIND nu sînt acceptate.

8. DEPANAREA PROGRAMELOR

Cind ati scris un program, acesta poate contine erori, care pot fi depistate în diferite faze ale procesului de compilare si executie. Erorile simple de sintaxa, cum sînt introducerea eronata a unui cuvînt cheie sau absenta unei parenteze, sînt depistate la introducerea liniilor de program, sau în prima parte a compilarii. Aceste tipuri de erori pot fi usor depistate si corectate. Erorile mai complexe nu pot fi detectate decît mai tîrziu în procesul de compilare, cînd sînt semnalate prin afisarea unui mesaj de eroare. In continuare se prezinta o lista cu mesaje de eroare si erorile posibil asociate:

27. Undefined Identifer (identificator nedefinit).

Intr-o expresie apare un identificator (nume de variabila sau functie), iar în program nu exista nici o instructiune prin care sa i se atribuie (asigneze) o valoare.

28. Wrong data type (tip de date eronat)

Tipul unei expresii sau a unei variabile nu este cel cerut, sau se foloseste numele unei subrutine sau a unui tablou acolo unde se cere o variabila simpla - sau viceversa.

Erori uzuale de acest tip sînt: amestecarea instructiunilor de tip de variabile sau a comenzilor, omiterea unor instructiuni obligatorii cum ar fi END.

29. Wrong statement order (succesiunea incorecta a instructiunilor)

Erori uzuale din aest tip sînt: amestecarea instructiunilor de declaratie cu instructiunile de executie, si/sau omiterea unor instructiuni obligatorii cum ar fi END.

30. Array Syntax error (eroare de sintaxa a tabloului)

Tipic aceasta eroare apare ca urmare a faptului ca un tablou a fost tratat ca variabila simpla sau invers.

31. Identifier declared twice (identidficator declarat de doua ori)

Aceslasi identificator apare de mai multe ori într-o declaratie, sau apare de doua ori în lista de parametri fictivi ai

unei subrutine.

32, Invalid item in list (lista de argumente invalida)
Tipic aceasta eroare apare cînd numele unui subprogram sau al unui argument fictiv apare în lista de variabile într-o instructiune COMMON, EQUIVALENCE, sau DATA.

33. Invalid equivalencing (echivalare invalida)
Aceasta eroare apare atunci cînd instructiunile COMMON si EQUIVALENCE cer ca aceeasi varabila sa fie memorata în doua locuri diferite.

34. Argument Mismatch (amestecarea argumentelor)
Argumentele actuale transmise unui subprogram nu corespund cu numarul sau tipul cerut de argumentele fictive.

35. Undeclared label (eticheta nedeclarata)
Eticheta unei instructiuni referita, nu este definita în cadrul aceluiasi program.

36. Array element out of bounds (element de tablou în afara limitelor tabloului)
Indicele unui element de tablou este în afara dimensiunilor specificate ale tabloului.

37. Label used twice (eticheta folosita de doua ori)
In cadrul unei unitati de program, doua instructiuni au aceeasi eticheta.

38. Invalid nesting (incubare invalida)
Apare în cazul în care într-o unitate de program buclele DO si blocurile IF sînt incluse (utilizate) incorect.

39. Syntax error (eroare de sintaxa)
Sintaxa programului nu corespunde cu specificatiile acestor instructiuni de utilizare.

4. Out of memory (s-a depasit capacitatea memoriei)

11. Integer out of range (numar întreg mai mare decît limita admisa)

ERORI CARE POT SA APARA IN FAZA DE EXECUTIE:

27. Wrong code in FORMAT statement (cod gresit în instructiunea FORMAT)
Tipul unui articol care urmeaza sa fie introdus sau afisat nu corespunde cu codul din FORMAT.

28. Invalid input (intrare incorecta)

29. Insufficient space to write (insuficient spatiul pentru scris)
Articolul care urmeaza sa fie afisat necesita un spatiu mai mare decît cel specificat în FORMAT.

30. Dummy procedure error

Tipul unei proceduri fictive sau al unuia din argumentele din lista sa de argumente nu corespunde cu cel al procedurii, sau cu argumentele sale curente.

31. Dummy Variable length error

Un tablou fictiv sau variabila de tip CHARACTER a fost declarata mai lunga decît lungimea corespunzatoare tabloului sau variabilei curente.

9. INCARCAREA SI UTILISAREA PROGRAMELOR DEMONSTRATIVE

Pentru a încarca primul program demonstrativ, PRIMES, introduceti 'LOAD' si porniti banda din pozitia corespunzátoare încárcárii primei párti a compilatorului.

Dupa ce a apárut mesajul 'Press required key', apásati tasta J si apoi ENTER pornind banda din pozitia convenabila pentru a încarca programul FORTRAN 'PRIMES' (program sursa).

Dupa aceasta apasati X pentru a-l compila, iar daca totul decurge normal, în partea de sus a ecranului vor apare semne neinteligibile si va fi emis un BEEP ca semn ca trebuie sa încarcati si a doua parte a compilatorului. Daca acesta s-a în-carcat normal, pe ecran vor apare mesajele prezentate în capitolul 6, PE CARE TREBUIE NEAPARAT SA LE NOTATI daca doriti sa salvati codul obiect pentru utilizari ulterioare. In aceastá situatie orice clapá ati apaása, calculatorul se va reseta, mentinînd NUMAI CODUL OBIECT DEASUPRA RAMTOP-ului. Comandati RANDOMIZE USR 63500 pentru a lansa în executie programul compilat si pe ecran vor fi afisate numerele prime între 1 si 1000.

Acum daca doriti sa încarcati de pe banda un alt program exemplu, trebuie sa încarcati din nou prima parte a compilatorului (care contine si editorul) si sa procedati ca în exemplul anterior.

Al doilea program demonstrativ, denumit FUNCT, contine rutinele necesare pentru a trasa graficul unei functii F(x), pentru evaluarea integralei acesteia si pentru rezolvarea ecuatiei F(x)=0 prin metode numerice. In program se exemplifica utilizarea rutinelor mentionate pentru cos(x) si EXP(-x*x/2)-x/2. Dupa putin exercitiu veti putea utiliza aceste rutine în programele dumneavoastra, pentru alte functii.

Al treilea program denumit LSFIT este un program de interpolare lineara prin metoda celor mai mici patrate, deci în mod firesc va cere sa introduceti numarul de puncte (perechi de date) si datele propriu-zise. Programul va determina dreapta care interpoleazá colectia de puncte, atît pentru Y în functie de X, cît sipentru X în functie de Y, si traseaza aceste drepte pe ecran. Numerele (date-le de intrare trebuiesc introduse prin canalul (calea/'stream') ?4, iar rezultatul va fi transmis pe canalul ?3 astfel ca utilizatorul TREBUIE CA IN PREALABIL SA DESCHIDA ACESTE CANALE pentru perifericele dorite: Daca de exemplu doriti sa introduceti datele de la claviatura si sa le obtineti la imprimanta, atunci INAINTE DE LANSAREA IN EXECUTIE A CODULUI OBIECT (programul compilat), trebuie sa comandati (în BASIC-ul calculatorului): OPEN3, 'P' (pentru iesirea pe imprimanta). Abia acum puteti lansa executia (cu RANDOMIZE USR 63500). Daca ati fi dorit

ca datele sa va fie prezentate pe ecran, (mai ales daca nu dispuneti de o imprimanta 'grafica'), în loc de OPEN3, 'P' trebuie comandat OPEN3, 'S'.

In cazul în care manipulati gresit, sau daca indicele unui tablou de elemente depaseste limitele acestuia, atunci vor apare mesaje de eroare (cap.8). Este foarte important ca expresiile din caractere utilizate ca identificatori de format sa fie corecte, deoarece în caz contrar, erorile nu pot fi detectate în timpul compilarii, si pot cauza derutarea programului în executie.

Compilatorul FORTRAN prezentat nu este deosebit de eficient pentru utilizatorul de virgula flotanta, deoarece din lipsa de spatiu, în acest caz sînt apelate rutinele din ROM.
El este însa deosebit de util celui care învata FORTRAN, sau celui care doreste sa verifice anumite subrutine degajînd astfel masinile 'mari' pe care vor rula în final programele complete.

IMPORTANT !

Edituta "TM" pune la dispozitia tuturor celor interesati întreaga gamă de manuale în limba română pentru calculatoare compatibile ZX Spectrum (TIM S, TIM S Plus, COBRA, HC 85, CIP, Jet) editate de firma "ALPHA Ltd" S.R.L. :

1.01 Limbajul **BASIC** pe înţelesul tuturor în 12 lectii
1.02 Documentatie **GENS** şi **MONS** (Asamblor-dezasamblor)
1.03 Documentatie limbaj **FORTH**
1.04 Documentatie **BETA BASIC 3.1** (Extensie BASIC)
1.05 Documentatie **BETA BASIC 3.1** (Rezumat)
1.06 Documentatie compilator **FORTRAN 77-S**
1.07 Documentatie editor de texte **TASWORD**
1.08 Documentatie compilator **BLAST**
1.09 Documentatie compilator **PASCAL HP4TM** (Rezumat)
1.10 Documentatie limbaj **C**
1.11 Memento timing cod maşină **Z80**
1.12 Documentatie **MEGA BASIC** (Extensie BASIC)
1.13 Documentatie **VU-CALC**
1.14 Manual **BASIC** avansati - conţinînd şi referiri la **COBRA**
1.15 Documentatie compilator **COLT**
1.16 Documentatie **MASTER - FILE** (sistem gestiune afaceri)
1.17 Documentatie limbaj **microPROLOG**
1.18 Documentatie limbaj **PASCAL HP4TM**
1.19 Documentatie sistem operare **CP/M** cu referire la calculatorul **COBRA**
1.20 Manual **ROM SPECTRUM** complet dezasamblat
1.21 Documentatie **LASER GENIUS** (pachet programe pentru lucrul în cod masină)
1.22 Cum să obtinem cît mai mult de la calculatorul nostru o carte cu programe si trucuri atît pentru începători cît si pentru avansati, în două variante:
a) Numai cartea, cu o parte din figuri în text
b) Cartea si o casetă demonstrativă, cu toate programele si figurile introduse
1.23 Construiţi singuri 20 de montaje electronice interfaţabile cu microcalculatorul Dvs